PowerInfer Fast Large Language Model Serving with a Consumer-grade GPU

论文信息

Title: PowerInfer Fast Large Language Model Serving with a Consumer-grade GPU

Authors: Yixin Song, Zeyu Mi, Haotong Xie and Haibo Chen(Shanghai Jiao Tong University)

Conference: SOSP '24, 15 November, 2024

研究背景

1. 在个人电脑和配备消费级 GPU 的本地平台上运行生成式大型语言模型（LLMs）的需求日益增长，这主要归因于对数据隐私的增强需求、模型定制化、降低推理成本以及专注于处理小批量数据时的低延迟。

2. 在消费级 GPU 上部署 LLMs 面临巨大的挑战，因为这些模型对内存的需求极大。

3. 解决此类内存问题的现有方法包括模型压缩和卸载。

LLM 推理架构

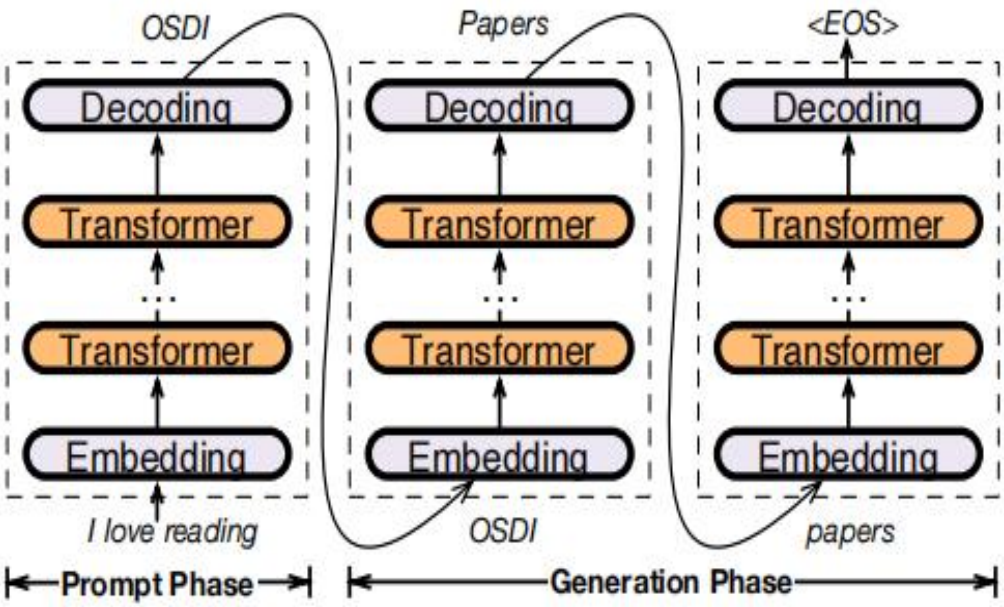


Figure 1: The inference procedure of an LLM.

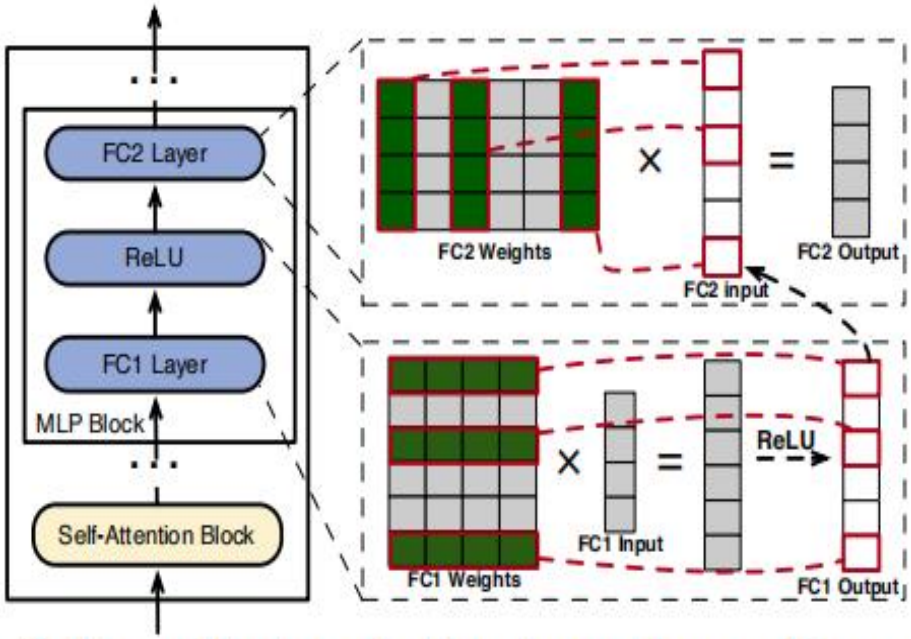
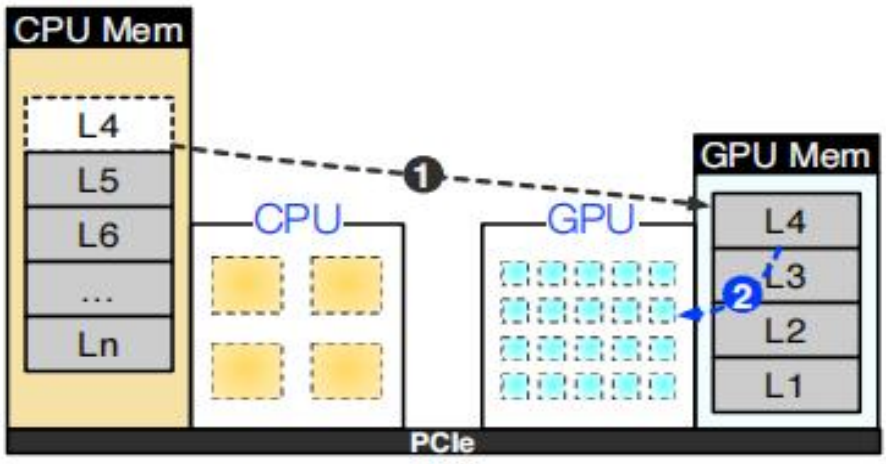


Figure 2: The architecture of a Transformer layer and how neurons are sparsely activated in FC1 and FC2 layers due to the ReLU function. The neurons that are activated are represented as green rows or columns encircled by red lines. The output vector from FC1 is then supplied to FC2 as its input vector.

基于 offloading 的 LLM 服务

GPU-centric offloading

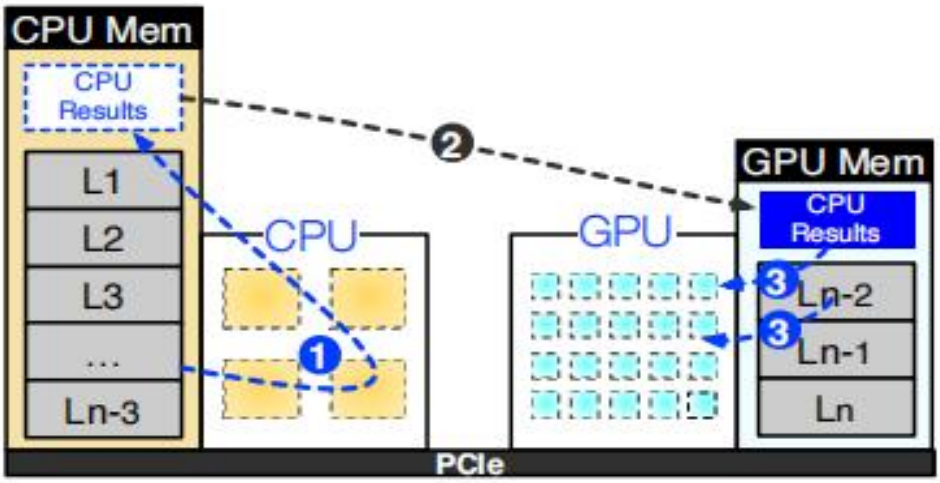
以 GPU 为中心的卸载：利用 CPU 内存来存储超出 GPU 容量的部分模型参数。



(a) GPU-Centric Offloading

Hybrid offloading

混合卸载：在 GPU 和 CPU 之间分配模型参数，在 Transformer 层级别进行分割。



(b) GPU-CPU Hybrid Offloading

动机

Insight-1: Power-law Activation

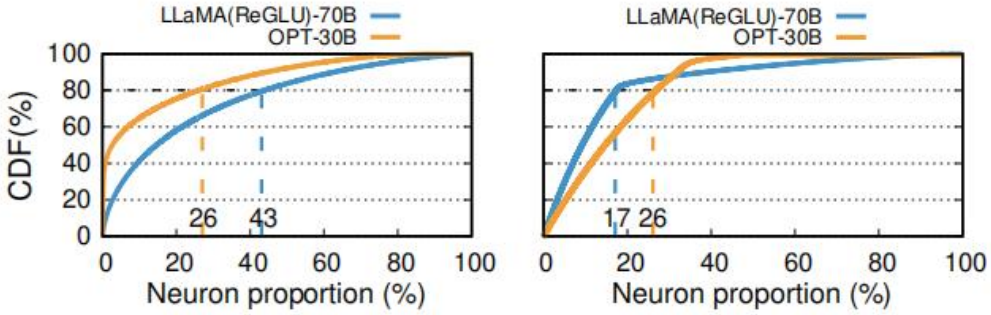
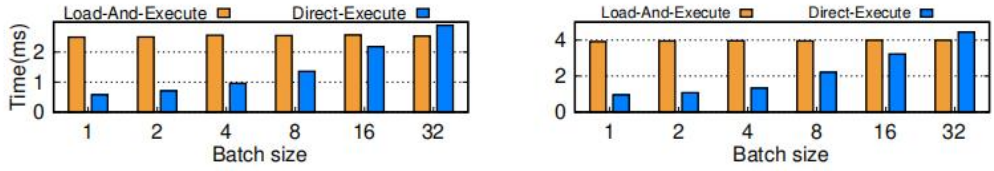


Figure 5: Cumulative distribution function (CDF) of neuron activation in OPT-30B and LLaMA(ReGLU)-70B. (a) CDF in a single MLP layer. (b) CDF across the entire model. The X-axis shows neuron proportion. The Y-axis represents the CDF of neuron activation.

Insight-2: Fast In-CPU Computation



(a) MLP layer (b) Attention layer

Figure 6: Comparison of execution time for load-then-execute versus direct-execute methods when 10% and 60% neuron weights of one MLP and attention layer in OPT-30B are CPU-resident. The X-axis shows input batch sizes, and the Y-axis measures execution time (ms). Load-then-execute involves transferring these neuron weights to GPU memory for computation, whereas direct-execute computes them directly on the CPU.

PowerInfer Overview

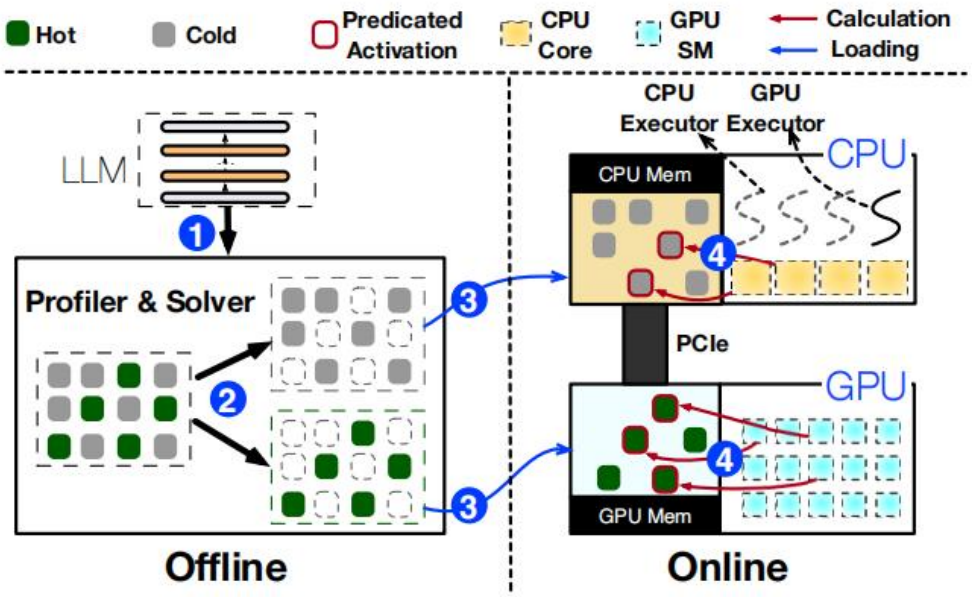


Figure 7: *The architecture overview and inference workflow of PowerInfer.*

神经元感知的推理引擎

Adaptive Sparsity Predictors

1. PowerInfer 中的在线推理引擎通过仅处理预测为激活的神经元来减少计算负载。

2. 然而，为资源有限的本地部署设计有效的预测器具有挑战性，需要在预测准确

性和模型大小之间取得平衡。

3.对于表现出显著偏斜的层，隐藏层大小逐渐减小，直到准确性降至95%以下。

Neuron Placement and Management

1.当离线求解器确定神经元放置策略时，PowerInfer根据该策略将模型加载到CPU和GPU内存中。

2.确保这些分割的神经元按正确顺序进行准确计算对于获得精确结果至关重要。

3.PowerInfer创建两个神经元表，一个位于CPU中，另一个位于GPU内存中。这些表将每个神经元与其在矩阵中的原始位置相关联。

4.这些神经元表所需的额外内存相对较少，对于像OPT-175B这样需要350GB存储的LLM，总内存仅为约9MB。

GPU-CPU Hybrid Execution

1.根据洞察-2，将激活的神经元传输到GPU的时间超过了直接在CPU上计算所需的时间。

2.在推理之前，PowerInfer构建一个计算有向无环图（DAG），其中每个节点代表一个计算LLM推理操作符，并将其存储在CPU内存的全局队列中。

3.在推理期间，由主机操作系统创建的pthreads类型的两种执行器分别管理CPU和GPU上的计算。

4.在激活的神经元在GPU和CPU之间分割的场景中，这两个处理单元之间的同步也变得至关重要。

Neuron-aware Operator

1.考虑到LLMs中的激活稀疏性，矩阵乘法操作可以绕过未激活的神经元及其权重，这需要使用稀疏操作符。

2.PowerInfer引入了神经元感知操作符，该操作符可直接在GPU和CPU上计算激活的神经元及其权重，而无需在运行时转换为密集格式。

神经元放置策略

Offline Profiling

1.为了收集每个神经元的运行时推理数据，PowerInfer将LLM部署到处理来自多个通用数据集（如C4和Wikipedia）的请求中。

2.此外，它在 GPU 上构建一个神经元信息表，用于跟踪每个神经元的激活次数。

Modeling of Neuron Placement

$$Maximize \quad t_i = \sum_{e \in \mathbb{N}} a_{ie} * v_e \forall i \in \{GPU\} \tag{2}$$

$$\sum_{i \in \mathbb{U}} a_{in} = 1 \quad \forall n \in \mathbb{N} \tag{3}$$

实验设置

硬件：

高端 PC：配备 Intel i9-13900K 处理器（八个物理核心，主频 5.4GHz）和 192GB 主机内存（内存带宽 67.2 GB/s）。该配置还包括 NVIDIA RTX 4090 GPU（24G），其内存带宽为 1TB/s，并通过 PCIe 4.0 接口（64GB/s 带宽）运行。

低端 PC：配备 Intel i7-12700K 处理器（八个物理核心，主频 4.9GHz），搭配 64GB 主机内存（内存带宽 38.4 GB/s）。它还包括 NVIDIA RTX 2080Ti GPU（11G），其内存带宽为 616GB/s，并使用 PCIe 3.0 接口（32GB/s 带宽）。

模型：我们使用 OPT 模型，参数从 6.7B 到 175B 不等，以及 Falcon(ReLU)-40B 和 LLaMA(ReGLU)-70B 模型。所有实验中的模型均使用 FP16 和 INT4 量化参数，中间激活使用 FP32。

工作负载：ChatGPT 提示和 Alpaca 数据集。

基线系统：llama.cpp、FlexGen 和 DejaVu。

关键指标：我们的主要评估指标是端到端生成速度，量化为每秒生成的平均词向量数（词向量/秒）。

End-to-End Performance

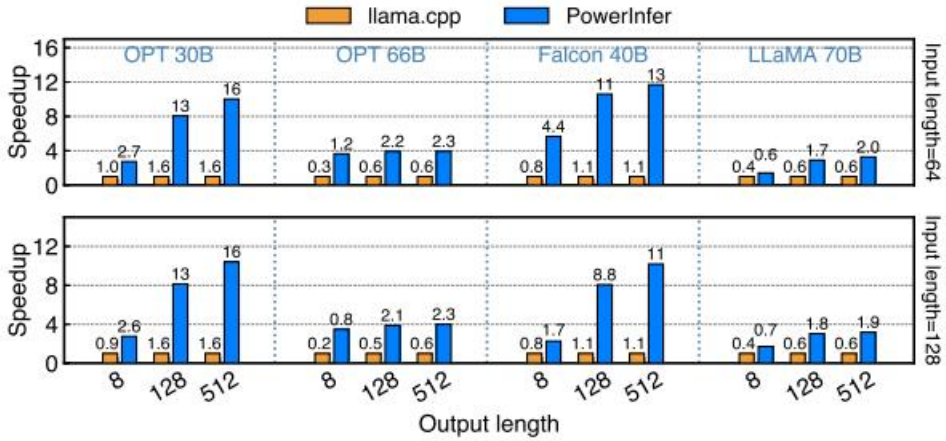


Figure 10: *Speedup of various models on PC-High in FP16 format.* The X axis indicates the output length. The Y axis represents the speedup compared with llama.cpp. The number above each bar indicates the end-to-end generation speed (tokens/s). The first row of the figure is configured with an input length of around 64, and the second row with an input length of approximately 128.

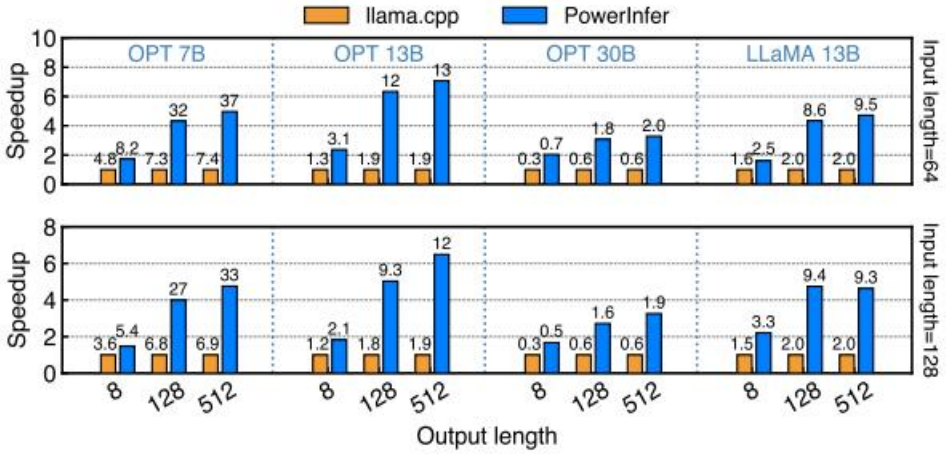
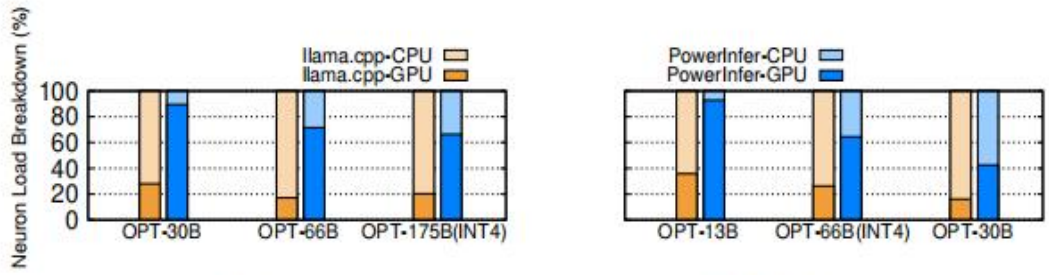


Figure 11: *Speedup of various models on PC-Low in FP16 format.* The X axis indicates the output length. The Y axis represents the speedup compared with llama.cpp. The number above each bar indicates the end-to-end generation speed (tokens/s). The first row of the figure is configured with an input length of around 64, and the second row with an input length of approximately 128.



(a) PC-High (b) PC-Low

Figure 12: *Neuron load distribution on CPU and GPU during inference*. The *yellow* block refers to llama.cpp, and *blue* block refers to PowerInfer.

Ablation Studies

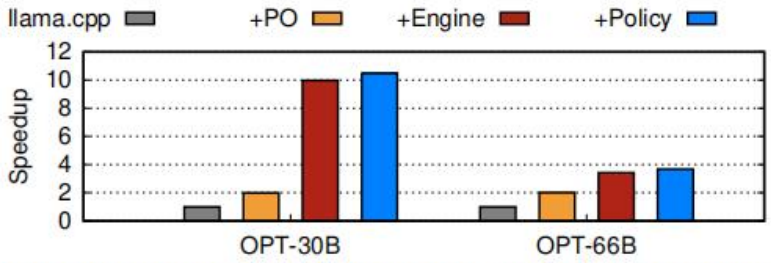


Figure 15: *Performance breakdown for each component of PowerInfer on PC-High.*

1. 首先，我们将 PowerInfer 的预测器和神经元感知操作符添加到 llama.cpp 中（标记为“+PO”），使 GPU 和 CPU 上仅能计算激活的神经元。

2. 在“+PO”的基础上，我们引入了 PowerInfer 的混合推理引擎（表示为“+Engine”）。

3. 最后一步是集成我们优化的策略（“+Policy”）。

结论

1. PowerInfer 是一个针对 LLMs 优化的快速推理系统，它利用了 LLM 推理中的局部性属性。

2. 它使用自适应预测器和神经元感知操作符进行神经元激活和计算稀疏性。

3. 与 llama.cpp 等系统相比，PowerInfer 实现了高达 11.69 倍更快的 LLM 推理速度，同时保持了准确性。

未来可以优化的方向

1. 研究 PowerInfer 的背景或动机是否合理，如果合理的话，可以尝试调查为什么没有人去做 swap。

2. 相比于基于 predictor 的方法，swap 的优势在于不需要考虑精度损失的问题，并且由于没有放置策略，swap 或许可以实现更快的推理速度。